CHAPTER – SEVEN

TESTING

7.1 Introduction

Testing is the process of evaluating a system and/or its components with the intent to find out whether it satisfies the stated requirements or not. This testing activity will give the actual and expected, and, the difference between actual and expected results. In other words testing means executing a system in order to identify any gaps, errors or missing requirements when compared with the actual or desired requirements.

Consider a situation where the client has asked to develop a banking application which should have functionality like cash withdrawal, funds transfer, account summary, SMS facility etc.

This application was developed and delivered to the client with a few bugs in it. The end users (customers of bank) will face problems due to bugs which were missed out without proper testing. The customer will be dissatisfied and sometimes sue the bank. To prevent these kind of scenarios testing is done.

Software testing can be stated as the process of validating and verifying a computer program/application/product:

- Meets the requirements which will guide the further design and development.
- To confirm whether the system works as expected.
- To confirm whether it can be implemented with the same characteristics.
- To confirm whether it satisfies the stakeholders needs

Testing is thus part of overall Quality Assurance (QA) from which we will be able to measure the quality of any software application. The software should meet its specified standards during its development cycle.

Software testing can be implemented at any time in the development process depending on the testing method employed. Test designing will start at the beginning of the project well before coding and thus will help to save a huge amount of the rework effort and resources.

7.2 Who Does Testing?

It depends on the process and the associated stakeholders of the project(s). Stakeholder can be referred to all the people who are part of the project being executed. Stakeholders are those people who are having an interest in the successful completion of the project. The important thing to remember is that the stakeholders should also have some say in defining the project objectives, since they are the people who will be affected by the outcome. When defining project stakeholders, the project manager and members of his/her team should carefully think through who will be the end users of the product. Project stakeholders usually include Customers/clients, company, project team including project manager, senior managers etc.

In the IT industry, large companies have teams with responsibilities to evaluate the software developed in the context of the stated requirements. Moreover, developers will also conduct testing which is called Unit Testing, which is explained later in the document. In most of the cases, the below professionals are involved in the testing of a system.

- Test analyst / Test Manager.
- Software Tester.
- Software Developer.
- End User.

Test analyst / Test Manager

During Test Planning phase, the activities carried out in Test Strategy (A Test Strategy document is a high level document normally developed by project manager. This document defines Testing Approach to clarify the major tasks and challenges of the test project) phase will be detailed and more application specific data will be collected. The above mentioned details along with the schedule of the project will be documented in the test plan.

Software Tester

The role of a software tester is mainly to write the test cases and perform integration/system testing.

End User

The Acceptance testing is done by the end user at the end of the project.

7.3 Levels Of Testing

Levels of testing include the different methodologies that can be used while conducting Software Testing. Following are the 2 aspects on which the Software Testing is carried out:

Functional Testing.

Non-Functional Testing.

7.3.1 Functional Testing:

Testing the application taking into consideration the business requirements. Functional testing is done using the functional specifications provided by the client or by using the design specifications like use cases provided by the clients.

- Functional Testing covers:
- Unit Testing.
- Integration Testing (Top Down and Bottom up Testing).
- System Testing.
- User Acceptance Testing.

Example

Consider the example of the banking ATM which will have different functionalities like cash withdrawal, deposit, balance inquiry, account summary etc.

Unit testing will include developing programs for each functionality mentioned and testing them.

Integration testing includes combining all the modules so that the information is passed correctly from one module to another. Say if an account has an initial amount of 1000 INR and if we deposit an amount of 2000 INR to the account then the balance inquiry should be an amount of 3000 INR.

Comprehensive black box testing of banking system with transactions initiated and validations performed on databases and reports generated while doing the account balance summary.

7.3.1.1 Unit Testing

This type of testing is performed by the developers before the product/application is handed over to the testing team to formally execute the test cases. Unit testing is performed by the software developers on the individual units of source code assigned to them. The developers use test data that is separate from the test data of the quality assurance team.

The goal of unit testing is to isolate each part of the program and show that the individual parts are working correctly in terms of requirements and functionality.

The limitations of Unit Testing are as follows:

Testing cannot catch each and every bug in an application. It is impossible to evaluate every execution path in a software application. The same is the case with unit testing.

There is a limit to the number of scenarios and test data that the developer can use to verify the source code. So after the developer has exhausted all options there is no other choice but to stop unit testing and combine the code segments with other units.

7.3.1.2 Integration Testing

The testing of combined parts of an application to determine if they function correctly together is Integration testing. There are two methods of doing Integration Testing; Bottom-up Integration testing and Top Down Integration testing which were already explained.

Bottom-up integration:

This type of testing begins with unit testing, followed by tests of progressively higher level combinations of units/programs called modules.

Top-Down integration:

In this testing, the highest/top level modules are first tested and then progressively lower-level modules are tested.

In a comprehensive software development environment, bottom-up testing is usually done first, followed by top-down testing. The process will finally conclude with multiple tests of the complete application, preferably in scenarios which are designed to mimic the environment which will be encountered in customers' computers, systems and network.

7.3.1.3 System Testing

This is the next level in testing in which we test the whole system. Once all the components are integrated, the entire application is tested rigorously to see whether it meets the quality standards. This type of testing is usually performed by a specialized testing team.

System testing is very important because of the following reasons:

1. System Testing is the first step in the Software Development Life Cycle, where the entire application is tested.

2. The application will be tested thoroughly to verify that it meets the functional and technical specifications.

3. The application is tested in a test environment which is very close to the production environment where the application will be deployed later.

4. System Testing enables us to test, verify and validate both the business requirements and the Applications Architecture.

7.3.1.4 Acceptance Testing

This is the most important type of testing as it is conducted by the Quality Assurance Team who will gauge whether the application meets the intended specifications and satisfies the client's requirements. The QA team (Alpha testing) will have a set of pre written scenarios and Test Cases that will be used to test the application. Alpha testing is a form of internal acceptance testing performed mainly by in-house software QA and testing teams. We also have Beta testing which is the final testing phase where companies release the software for few external user groups outside the company test teams or employees.

More ideas will be shared about the application and more tests can be performed on it to determine its accuracy and the reasons why the project was initiated. Acceptance testing is done not only to point out simple spelling mistakes, cosmetic errors or Interface gaps, but also to point out any bugs in the application that will result in system crashes or major errors in the application at a later stage.

By performing acceptance tests on an application, the testing team will get to know how the application will perform in production environment. There may also be legal and contractual requirements for acceptance of the system. In such cases the application should satisfy the above requirements before it is accepted by the client.

7.3.2 Non-Functional Testing:

Testing the application against client's and performance requirement. Non-Functioning testing is done based on the requirements and test scenarios given by the client.

- Non-Functional Testing covers:
- Load and Performance Testing
- Stress & Volume Testing
- Compatibility & Migration Testing
- Data Conversion Testing
- Operational Readiness Testing
- Security Testing
- Performance testing

7.4 Test Techniques

7.4.1 Black Box Testing

It is testing without knowledge of the internal workings of the item being tested. For example, when black box testing is applied to software engineering, the tester would only know the "legal" inputs and what the expected outputs should be, but not how the program actually arrives at those outputs. It is because of this that black box testing can be considered testing with respect to the specifications, no other knowledge of the program is necessary. For this reason, the tester and the programmer can be independent of one another, avoiding programmer bias toward his own work. For this testing, test groups are often used, "Test groups are sometimes called professional idiots...people who are good at designing incorrect data." Also, due to the nature of the black box testing, the test planning can begin as soon as the specifications are written. The opposite of this would be glass box testing, where test data are derived from direct examination of the code to be tested. For glass box testing, the test cases cannot be determined until the code has actually been written. Both of these testing techniques have advantages and disadvantages, but when combined, they help to ensure thorough testing of the product.

7.4.1.1 Advantages of Black Box Testing

- More effective on larger units of code than glass box testing
- Tester needs no knowledge of implementation, including specific programming languages
- Tester and programmer are independent of each other
- Tests are done from a user's point of view
- It will help to expose any ambiguities or inconsistencies in the specifications
- Test cases can be designed as soon as the specifications are complete.

7.4.1.2 Disadvantages of Black Box Testing

- Only a small number of possible inputs can actually be tested, to test every possible input stream would take nearly forever
- Without clear and concise specifications, test cases are hard to design
- There may be unnecessary repetition of test inputs if the tester is not informed of test cases the programmer has already tried
- May leave many program paths untested

7.4.2 White Box Testing

White box testing is performed based on the knowledge of *how* the system is implemented. White box testing includes analyzing data flow, control flow, information flow, coding practices, and exception and error handling within the system, to test the intended and unintended software behavior. White box testing can be performed to validate whether code implementation follows intended design, to validate implemented security functionality, and to uncover exploitable vulnerabilities.

White box testing requires access to the source code. Though white box testing can be performed any time in the life cycle after the code is developed, it is a good practice to perform white box testing during the unit testing phase.

The general outline of the white box testing process is as follows:

- Perform risk analysis to guide the whole testing process.
- Develop a test strategy that defines what testing activities are needed to accomplish testing goals.

- Develop a detailed test plan that organizes the subsequent testing process.
- Prepare the test environment for test execution.
- Execute test cases and communicate results.
- Prepare a report.

For a complete software examination, both white box and black box tests are required.

Testing is applied to find that how much our program is efficient. Testing is critical phase of software quality assurance. It indicates the review of specification, design & code generation. After completing source code software must be tested to find some uncover error, before delivered it.

Thus, a series of test cases has to be designed that have a high likelihood of finding an error. To accomplish this task, software testing methods are used.

These methods are:

- We exercise the internal logic of the software component.
- We exercise program's input & output domains, thus uncovering error in program function, behaviour, & performance.

7.5 Test Cases

A test case is a set of conditions under which tester will determine whether an application, software system or one of its features are working as it was originally established for it to do software system or one of its features are working as it was originally established for it to do testing.

- Test results for decision tree model

Test Case Id	Actual.values	Predicted.Values
1	N	Y
2	Y	Y
3	Y	Y
4	Y	Y
5	Y	Y
6	Y	Y
7	N	N
8	Y	Y
9	N	Y
10	Y	Y
11	Y	Y
12	N	Y
13	Y	Y
14	Y	Y
15	N	N

- Test results for logistic regression model

Test Case Id	Actual.values	Predicted.Values
1	N	Y
2	Y	Y
3	Y	Y
4	Y	Y
5	Y	Y
6	Y	Y
7	N	N
8	Y	Y
9	N	N
10	Y	Y
11	Y	Y
12	N	Y
13	Y	Y
14	Y	Y
15	N	N